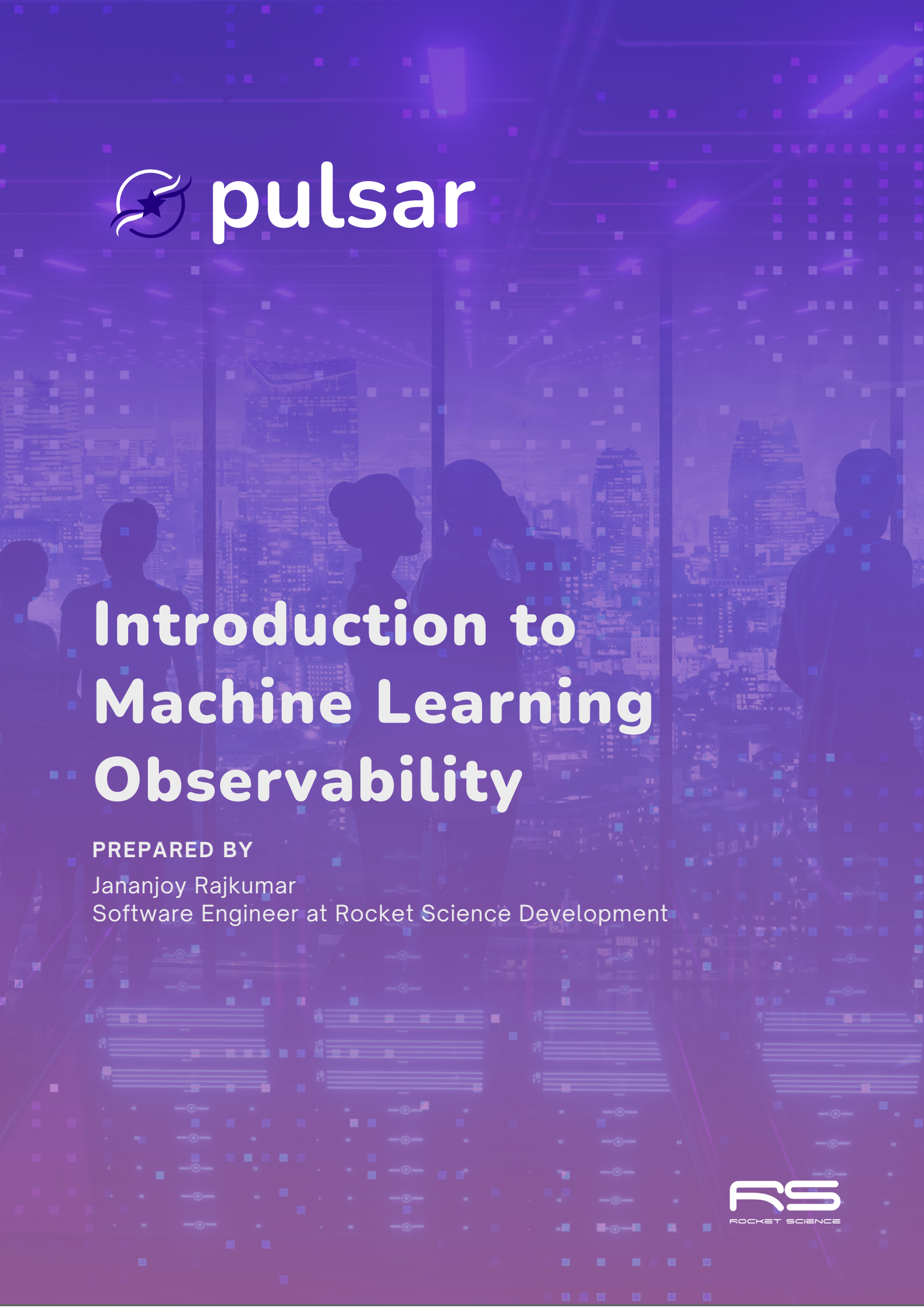

Introduction to Machine Learning Observability

PREPARED BY

Jananjoy Rajkumar
Software Engineer at Rocket Science Development

ROCKET SCIENCE

pulsar

1. INTRODUCTION

In today's world, we interact with Artificial Intelligence (AI) in some form or another on a daily basis. Businesses, on the other hand, are increasingly utilizing Machine learning (ML) models, which are the most common types of AI, to enhance their operational efficiency and decision-making capabilities. However, the deployment of ML models in business operations come with various technical challenges, such as performance, accuracy, and scalability. These ML models now serve a wide range of use cases in many different forms and modalities (tabular, time series, text, image, video, and audio). Furthermore, these models handle vast amounts of data, which can be either delay-sensitive or reliable sensitive, to deploy in a cloud or distributed platform. This mode of deployment and execution adds complexity to the system, which can negatively impact its performance.

2. ML OBSERVABILITY

The software development process with ML components is more complex than the traditional approach. Such a development process needs to monitor ML model performance, data quality, and production code for any changes and variations in the ML production life cycle. However, the ML monitoring approach of tracking the model's performance and understanding the issues associated with the model is not sufficient to drill down to the complex problem of today's ML model performance issues. This approach might resolve surface issues but not drill down to the root cause of problems.

To address this problem, we introduce ML Observability, which is a holistic approach to gaining more insights into the ML model's behavior, data, and performance. ML Observability measures the system's internal states by examining the ML model's output. It becomes an integral part of the ML life cycle to gain better control over complex systems by identifying the root causes of problems.

pulsar

The main objectives of implementing ML observability in the ML life cycle are:

- To gain a better understanding of ML model's decisions.
- To perform ML mode performance measurement.
- To Identify the root causes of model performance issues and aiding in troubleshooting.

Today, ML models are used across all industries and by companies for their mission-critical tasks and operations, as well as to provide value to their customers. Businesses invest heavily in building, testing, and experimenting with their models to improve their operational efficiency and decision-making capabilities. However, according to recent surveys, over 84% of data scientists and ML engineers face challenges in resolving problems related to ML models. To address this pain point, the implementation of ML Observability requires a systematic approach to detect and diagnose model problems, thereby facilitating efficient performance measurement. The right ML observability metrics and checklists not only detect emerging issues but also provide deeper and proactive introspection to diagnose the root cause of problems before they significantly impact the business or its customers. With the right problem-solving strategy, ML teams can tackle problems by examining all possible sources of the problem, such as data, models, and code.

In our Pulsar.ML project, we monitor ML model's input data and model output, and their evolution over time with various performance metrics. Our core functionality also includes automatic monitoring of the ML model by relevant stakeholders when various performance metrics reach important thresholds. Additionally, we implement a systematic methodology to measure the performance of ML models using ML observability guidelines and protocols.

pulsar

3. ML OBSERVABILITY CHECKLIST IN PULSAR.ML PROJECT

In this section, we explain the recommended checklist implemented in our Pulsar.ML project development life cycle to trace the root causes related to performance issues.

3.1 Model Lineage, Validation & Comparison

Model versioning tracks the dependencies that affect ML model performance. We also test multiple models in various ML pipelines to tune parameters and hyperparameters, as well as to measure model accuracy. We perform ML model lineage to track the data flow over time including where the data originated, how it has changed and its ultimate destination within the data pipeline. We also record the data throughout the ML lifecycle, including source information and any data transformations applied during any ELT (Extract, Load, Transform) and ETL (Extract, Transform, Load) processes. Finally, we perform pre-launch model validation as an exploratory test to establish a baseline for model performance.

3.2 Data Quality Monitoring & Troubleshooting

We monitor the production model with respect to input data and model outputs using a configurable baseline setup. Our software can detect anomalous behavior, including outlier detection on predictions. The Pulsar.ML project implements configurable real-time statistics on features and predictions (such as min, max, median, mean, and standard deviation) both in aggregate and by cohorts.

3.3 Drift Monitoring & Troubleshooting

The Pulsar.ML project includes drift monitoring as one of its key features for any flexible dataset. We compare the training and production distributions using a configurable baseline setup to detect drift. This includes overall production drift detection for concept, data, and model. In addition, we troubleshoot model drift by drilling down into feature drift to measure performance.

pulsar

3.4 Performance Monitoring & Troubleshooting

We monitor the ground truth by combining predictions with delayed response label data, using configurable baselines that support both production and pre-production environments. Constant and dynamic thresholds are used to monitor production models. Pulsar.ML project provides dynamic cohort analysis and segmentation of predictions and can surface performance problems automatically by feature, value, or cohort, without requiring the user to write SQL queries. Our interactive dashboard makes it easy for all stakeholders, including non-technical personnel, to understand the problem. We currently have the ability to compare model performance metrics, such as ROC-AUC, accuracy, precision, and R-squared, from the trained model to the production model.

3.5 Explainability

We implement explainability by providing the ability to view the feature importance of the top features. We support global, cohort, and local explainability to assist in all stages of the ML lifecycle.

3.6 Business Impact Analysis

We define custom User Defined Functions (UDF) to establish a baseline for model performance with business metrics. We also analyze thresholds for probability-based decision models dynamically and compare pre-production models to the current production models.

3.7 Integration Functionality

We develop Pulsar.ML project with support for loosely coupled distributed components, including a Software-as-a-Service (SaaS) deployment model, on-premises deployment, and hybrid deployments. Pulsar.ML can automatically infer the model type and calculate metrics for monitoring. It also provides interfaces for importing and exporting data from or to external data sources. Instead of providing an end-to-end hosting and serving system, we focus on implementing the best guidelines and techniques for ML model monitoring and observability, ensuring product depth and user customization.

3.8 UI/UX Experience

Our dashboard is designed with flexible and customizable features, allowing technical and non-technical stakeholders to easily visualize performance metrics. The dashboard also enables monitoring of any changes in models, code, and input data.

pulsar

3.9 Scalable to Meet Current & Future Analytics Complexity

We are currently working on implementing additional functionality to enhance scalability. Our scalability features are designed to handle large-scale analytic workloads for deployment. This includes support for load testing large numbers of features and large-scale deployments.

4. CONCLUSION

Maintaining a healthy model in production can be challenging, as ML models are dynamic, and their performance is heavily reliant on data quality and training data. Besides, High rate of volatility in data distribution (for time series data) can have a significant impact on model performance. Therefore, the implementation of ML model monitoring techniques alone is not sufficient enough for effectively measured the performance of the complex ML models .

ML observability provides a deep understanding of a model's data and performance throughout its lifecycle.It helps ML practitioners to trace the root cause of the problem and also explain the reason of misbehaving the model in the different way. Besides, It helps to identify production issues, identify technical cause of model performance issues, validate models, compare model performances and technical gaps in training data.

In conclusion, we recommend to elevate ML efforts with observability instead of relying on guesswork. ML observability tools assist practitioners to resolve performance problems throughout ML lifecycle. Always keep in mind that they are not a magic bullet. Besides, commitment from higher-level stakeholders in the organization and maturing of processes are also crucial to get the maximum benefits out of it.